Autonomous Robots, 2, 1–20 (1992)

Improved Rover State Estimation in Challenging Terrain

BRIAN D. HOFFMAN
Dept. of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02215

ERIC T. BAUMGARTNER
Science and Technology Development Section, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

TERRY HUNTSBERGER
Intelligent Systems Laboratory, Department of Computer Science, University of South Carolina, Columbia, SC 29208

PAUL S. SCHENKER
Science and Technology Development Section, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

Received April 29, 1991; Revised October 11, 1991

Editors: B. Bhanu

Abstract. Given ambitious mission objectives and long delay times between command-uplink/data-downlink sessions, increased autonomy is required for planetary rovers. Specifically NASA's planned 2003 and 2005 Mars rover missions must incorporate increased autonomy if their desired mission goals are to be realized. Increased autonomy, including autonomous path planning and navigation to user designated goals, relies on good quality estimates of the rover's state, e.g. its position and orientation relative to some initial reference frame. The challenging terrain over which the rover will necessarily traverse tends to seriously degrade a dead-reckoned state estimate, given severe wheel slip and/or interaction with obstacles. We present the implementation of a complete rover navigation system. First, the system is able to adaptively construct semi-sparse terrain maps based on the current ground texture and distances to possible nearby obstacles. Second, the rover is able to match successively constructed terrain maps to obtain a vision-based state estimate, which can then be fused with dead reckoned wheel odometry to obtain a much improved state estimate. Finally the rover makes use of this state estimate to perform autonomous real-time path planning and navigation to user designated goals. Reactive obstacle avoidance is also implemented for roaming an environment in the absence of a user designated goal. The system is demonstrated in soft soil and relatively dense rock fields, achieving state estimates much improved with respect to dead reckoning alone (e.g. 0.38 m mean absolute error vs. 1.34 m), and successfully navigating in multiple trials to user designated goals.

Keywords: Planetary rovers, range map registration, state estimation, vision-based navigation, real-time path planning

1. Introduction

The current planned objectives for NASA's 2003 and 2005 rover missions to Mars include traverses totalling tens of kilometers, over time frames spanning several months to a year or more. During these long distance traverses, various science functions are to be performed, including sample acquisition and cache, spectroscopic imaging, micro-camera imaging, and sample abrasion. In order to adequately achieve these ambitious desired mission objectives within the specified time frames, increased autonomy must be introduced and incorporated into NASA's 2003 and 2005 rover platforms. Given the long delay times between command-uplink/data-downlink sessions, increased levels of autonomy are necessary as teleoperation is infeasible, and high average ground speeds are desired to maximize the amount of terrain covered per unit time.

Of paramount importance are the robustness and computational efficiency of the autonomous behaviors. Not only must these algorithms function with repeatedly high performance in varying terrains and lighting conditions, but also work well given the computational limitations imposed by radiation hardened processor requirements and power constraints. Additionally, these behaviors must take into account and be capable of dealing with the possiblities of sensor and actuator degradation over time, both during transit, landing, and surface traverse.

Thus, we desire that the rover be capable of robustly and autonomously completing the following tasks:

- obstacle avoidance
- state estimation
- real time path planning to a user designated goal

For the task of obstacle avoidance, we desire that the rover be capable of distinguishing hazardous terrain features with a minimum of false positives or undetected hazards, with the former requiring the rover to perform excessive, time consuming, avoidance maneuvers, and the latter being directly hazardous to the health and well being of the rover.

Our implementation of autonomous obstacle avoidance, using a texture-based interest operator [6] coupled with stereo correlation and triangulation, is adaptive to varying surface and lighting conditions, as well as being computationally efficient.

In soft soil wheel slip is significant, resulting in degradation of the dead-reckoned vehicle state estimate. Likewise, in terrain with dense rock fields, obstacle climbing and numerous rotations in obstacle avoidance maneuvers significantly degrade the dead-reckoned vehicle estimate.

Thus, for the task of state estimation, we implement an extended Kalman filter [2] to fuse together two estimates of the rover's change in state. The first estimate is visually based, using terrain maps generated at each frame during the rover's traverse to derive the rover's between-frame rotation and translation [27]. The second estimate is the dead reckoned estimate, using the rover's wheel odometry. The combination of the two measurements gives us a much more accurate state estimate than relying on either measurement alone.

Finally, using our current Kalman filtered state estimate, we replan our desired path to the user designated goal at each point, using a real-time 2D potential flow based path planning methodology [7], addressing the problem of real-time path planning to a user designated goal. These approaches have been experimentally verified using NASA/JPL's Lightweight Survivable Rover (LSR-1), pictured in Figure 12.

Much work has been done in range map generation, some using "texture" features such as corners or other interest point identification methods [3] [17]. Some generation algorithms use parallel approaches [8]. Given the computational limitations imposed by power and radiation hardened processor requirements, we have developed a computationally efficient semi-sparse range map generation algorithm which is adaptive to both distance and texture.

Likewise, much work has been done in the area of mobile vehicle state estimation, often using vision [2] [29] [17] [11] [19] [9] [14], or using statistical estimation techniques to fuse together measurements of the rover state from multiple sensors [2] [17] [22]. However, experimental validation of statistical state estimation approaches such as Kalman filtering for mobile robotics generally

only been performed in laboratory settings, benign terrain, [2] [17], [14], [4] or only in simulation [22]. Some navigation performance evaluation has been pursued [16], however state estimation during these trials relied on dead reckoning only. We add experimental validation in very difficult terrain, as might be experienced by a rover performing traverses on the surface of Mars, and show the feasibility of such a sensor fusion approach to state estimation.

It is important to note that the state estimation framework is not only valid for vision-based measurements as described in this paper, but also for other sensor inputs as well. In fact researchers have investigated the use of inertial navigation sensors (INS) in rover systems [1], [21]. The fusion of vision-based map registration with wheel odometry provides an important capability for determining rover state estimates that can only be enhanced by the inclusion of additional independent measurement sources such as INS systems.

Potential flow based path planning has been demonstrated extensively in simulation, [7], [12], [5], [20], [13], (see [7] for more references). Here, we demonstrate experimental validation of real-time path planning using harmonic potentials in dynamic environments. Additionally, the environment consists of soft soil, and relatively dense rock fields, validating the potential flow based path planning approach in complex and challenging environments.

Finally, we show the unification of these techniques in system level trials. This includes adaptive range map generation, successive range map registration, Kalman filtering of vision and dead reckoning for improved state estimation, and real time path planning using potential flow-based methodologies. Path planning relies on the state estimate, which relies on Kalman filtering of the registered range maps, which relies on range map registration, all within an integrated system. However, this also validates each of the system components, which could of course be used separately in another system.

The following section presents the implementation of our vision-based rover state estimator. This includes a discussion of our high-speed texture-based adaptive range map generation algorithm in section 2.1. It also includes an overview of our implementation of a range map registration algorithm [27] in section 2.2. We detail our approach to rover state estimation via Kalman filtering of vision (the output of the range map registration algorithm) and dead reckoning in section 3.

An overview of the experimental system is presented in section 4, with the results being presented in section 5. As part of the results, we discuss two navigation methodologies, reactive obstacle avoidance in section 5.1 and real-time path planning to a user designated goal in section 5.2. Finally, we present conclusions and plans for future efforts in section 6.

2. Adaptive Range Map Generation and Vison-Based State Estimation

To address the problem of robust and computationally efficient obstacle avoidance, we present the novel application of a texture-based interest operator in conjunction with stereo correlation and triangulation to generate range maps, adaptive in refinement to both texture and distance.

2.1. Adaptive Range Map Generation

To adaptively generate a terrain map, we begin by acquiring a stereo image pair from the rover navigation cameras. A sample image pair may be seen in figure 1.

2.1.1. Wavelet Segmentation

Once the images are acquired, we compute the wavelet decomposition of each image using the standard two dimensional Harr wavelet basis [24]. The wavelet transform gives a multi-resolution view of an image with an average channel and three detail channels corresponding to horizontal (H), vertical (V), and diagonal (D) features. The wavelet decompositions are thresholded based on pixel value from 0 to 255. In our implementation, we use a threshold value of 4 which gives us adequate results. A representative wavelet decomposition from one of the stereo images is shown in figure 2.

The "ground" texture is then computed in several known locations at the bottom of each image,

Fig. 1. Stereo camera pair with ground texture computation location highlighted.

between the rover wheels, as shown in figure 1 using a texture operator on the wavelet decompositions. We assume that the portion of the stereo images close to and in between the wheels of the rover are free of obstacles. The output of this operation is a texture signature, TS, in the space defined by the H, V, and D channels of the wavelet decomposition [6] and, for each channel, is defined as

$$TS = \sum_{i=0}^{l} \frac{\frac{\sum_{j=-n/2}^{n/2} \sum_{k=-m/2}^{m/2} |W(a+j,b+k)|}{(nm)}}{l} \qquad (1)$$

where the parameter l refers to the level to which the wavelet decomposition, W, has been computed. The summation in the numerator of equation (1) effectively gives the texture energy of a local neighborhood at a given level of resolution [15]. In our implementation, we use $l = 2$, as can be seen in figure 2. Thus, for each channel (H,V,D) in the wavelet decomposition, we sum the contributions to the texture signature from each level of the decomposition, by summing the absolute value of the thresholded wavelet coefficients over a window of size $n \times m$ centered on (a, b) as indicated by equation (1). The values of n and m are dependent upon the current level, and the coordinates (a, b) are the image coordinates (x, y) in the wavelet coefficient space at level l and channel (H, V, D). In our implementation, $n = m = 8$ at $l = 0$ and $n = m = 4$ at $l = 1$, and $n = m = 2$ at $l = 2$.

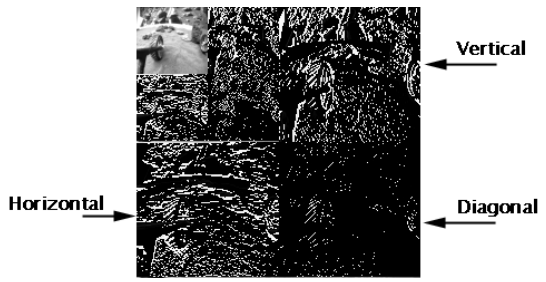

Fig. 2. Wavelet transform showing horizontal, vertical and diagonal channels for a $l = 2$ decomposition with the $l = 1$ decomposition results shown in the upper left quadrant.

Once we have an average H,D,V texture signature which we assume is the ground, we search the wavelet coefficient space of the left image with a stepsize of 16 pixels, in an area where we expect to find obstacles (i.e. slightly ahead of the rover wheels), looking for points which do not match the average ground texture signature which has been previously computed. These "interesting" points are marked, as shown in figure 3. A range map computed using a stepsize of 16 will be referred to as a "Level 16" range map. Similarly, a range map computed with a stepsize of 8 will be referred to as a "Level 8" range map, and so on.

This method of texture-based feature point selection is dependent on the initial "ground" signatures, in that "interesting" points are ones whose signatures fall outside of the bandwidth of the safe points. As such, it is an adaptive algorithm that is robust with relation to local changes in the terrain. The density of the distribution of feature points is directly tied to the search radius (stepsize) in the wavelet coefficient space. If a dense map is needed for relatively rough terrain, a smaller stepsize is required. We are currently investigating adaptive stepsize generation based on the density of feature points returned by the algorihm (see section 2.1.4).

2.1.2. Correlation

We then attempt to correlate the interesting points found in the left stereo image with their corresponding points in the right stereo image, using the fundamental matrix [28] to reduce this search from a 2D problem to a 1D problem. Although we are using 120° field of view lenses on the stereo camera pair, we have found that the error in using the fundamental matrix to compute a line of correspondance (rather than computing

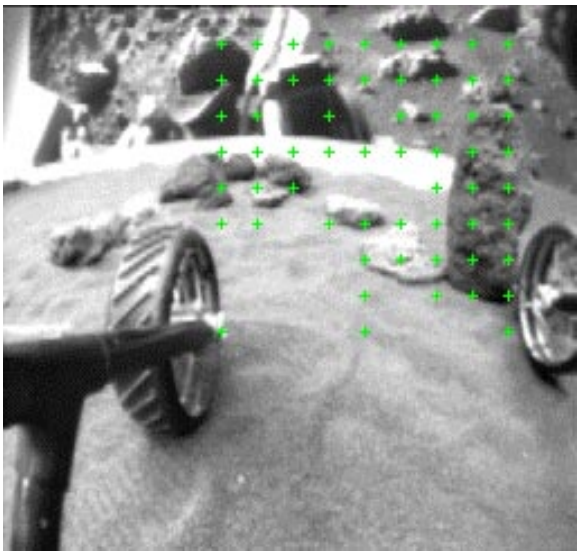

Fig. 3. Points of interest, Level 16.

a curve of correspondance as would be expected for such highly distorted images) is on the order of approximately one pixel [31]. This experimental result is corroborated by the investigation performed in [30].

In correlating, we search along the epipolar line of left image point $\mathbf{m_1}$ given by $\mathbf{Fm_1}$, where $\mathbf{F}$ is the 3×3 fundamental matrix, and $\mathbf{m_1}$ is a vector in the form $[\ x\ y\ 1.0\ ]^T$, where x and y are the column and row pixel locations, respectively. We assume that the cameras are reasonably aligned to the same horizontal axis, and thus limit our search vertically to ± 10 rows. Figure 4 details a left image point of interest and its corresponding epipolar line and search band. To account for possible errors in the epipolar geometry, we also search left and right at each point on the epipolar line by ± 8 columns. However, if we use cameras which can reasonably be modeled as orthographic, then this error is greatly reduced and, correspondingly, the number of columns to the right or left of the epiolar line we bother searching is also reduced. For our stereo cameras, the $\mathbf{F}$ matrix is

$$\begin{bmatrix} -8.907e-06 & 1.159e-03 & -7.025e-02 \\ -1.155e-03 & 2.457e-04 & 1.966e-01 \\ 6.521e-02 & -2.082e-01 & 9.532e-01 \end{bmatrix}.$$

Fig. 4. Correlation using the fundamental matrix.

A valid match between a left image point and its corresponding point in the right image is defined by the greatest correlation score found during the search along the right epipolar line, larger than some threshold. To compute the correlation scores between points in the left and right image, we employ the standard sum minus average divided by standard deviation over an $n \times m$ box in each image.

In this normalized definition of correlation, the maximum correlation value is $+1$ (identical) and the minimum value is -1 (totally uncorrelated). In our implementation the correlation threshold is chosen to be 0.80, following [28]. Likewise, n and m are chosen to be 7, which has become the defacto standard correlation window size [8]. The correlation score between a left image point $\mathbf{m_1}$ and a right image point $\mathbf{m_2}$ is defined as

$$Score(\mathbf{m_1}, \mathbf{m_2}) = \frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} \begin{matrix}[I_1(u_1+i,v_1+j) - \overline{I_1(u1,v1)}] \times \\ [I_2(u_2+i,v_2+j) - \overline{I_2(u2,v2)}]\end{matrix}}{(2n+1)(2m+1)\sqrt{\sigma^2(I_1) \times \sigma^2(I_2)}} \quad (2)$$

where $\overline{I_k(u,v)}$ is defined by equation (3) and is the average image intensity over a window of size $(2n+1) \times (2m+1)$ centered on (u,v) in image I_k, where $k = 1, 2$. The average image intensity is defined as

$$\overline{I_k(u,v)} = \sum_{i=-n}^{n} \sum_{j=-m}^{m} \frac{I_k(u+i, v+j)}{[(2n+1)(2m+1)]} \quad (3)$$

The variable $\sigma(I_k)$, in equation (1), is defined by

$$\sigma(I_k) = \sqrt{\frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} I_k^2(u,v) - \overline{I_k(u,v)}}{(2n+1)(2m+1)}} \quad (4)$$

and is the standard deviation in the image intensity over a window of size $(2n+1) \times (2m+1)$ centered on (u, v) in image I_k, where $k = 1, 2$.

2.1.3. Subpixel Correlation

Once we have an initial point match between the left and right images, we compute the subpixel location of the matching point in the right image, which is obtained by fitting a parabola to the pixel-level correlation scores in both the row and column directions, as given by [26]

$$\begin{aligned} \Delta col &= \frac{S_{col-} - S_{col+}}{2(S_{col-} + S_{col+} - 2S_{col})} \\ \Delta row &= \frac{S_{row-} - S_{row+}}{2(S_{row-} + S_{row+} - 2S_{row})} \end{aligned} \quad (5)$$

where S_{row} and S_{col} refer to the correlation score between a point in the left image $\mathbf{m_1}$ and a point in the right image $\mathbf{m_2}$ at position (col, row), where the maximum pixel level correlation score has been found. S_{col-} is the score between $\mathbf{m_1}$ and the point in the right image one column to the left of $\mathbf{m_2}$. S_{row-}, S_{col+} and S_{row+} are computed similarly.

The right image row and column are then set equal to their integer value found during the pixel level correlation, plus the delta row and column found during the subpixel correlation. This procedure greatly improves the smoothness of the derived range map, with little extra computation.

Using the set of matching image points computed to subpixel precision, we then triangulate to obtain 3D position information, using metrically calibrated camera models that take into account radial lens distortion [10].

2.1.4. Adaptive Mesh Refinement

Using the computed 3D position information, we can further iteratively improve the density of our range map by returning to those interesting points computed in the previous iteration that are found to be within a threshold "danger distance" from the rover.

Searching the image over an area of $(Stepsize_{i-1})^2$ centered on each interesting point using a stepsize of $(Stepsize_{i-1})/2$, we find a new set of "interesting points" based on texture and compute their 3D position information, as can be seen in figure 5.

In this fashion we perform what we have dubbed "adaptive mesh refinement", being somewhat akin to the process of adaptive mesh refinement in finite element analysis, in that we only bother computing the range map in areas where we care about what we are going to find, i.e. areas that are both texturally interesting and "dangerous", or close to the rover.

The density of the final range map is user specified. The initial Level 16 range map is computed with a pixel stepsize of 16, which at 2 meters from the rover corresponds to approximately 26 cm between adjacent range map points for our 256 x 243 stereo image pairs. Level 8 is the next step up in density, with Levels 4, 2, and finally, 1, becoming increasingly more dense.

Figure 6 shows how the range map has been computed adaptively such that the areas corresponding to the "ground" in texture are not computed. Likewise, it can be seen in the upper left corner that initially interesting points texturally are not further refined, as they are too far away to be either accurate in distance or immediately dangerous to the rover.

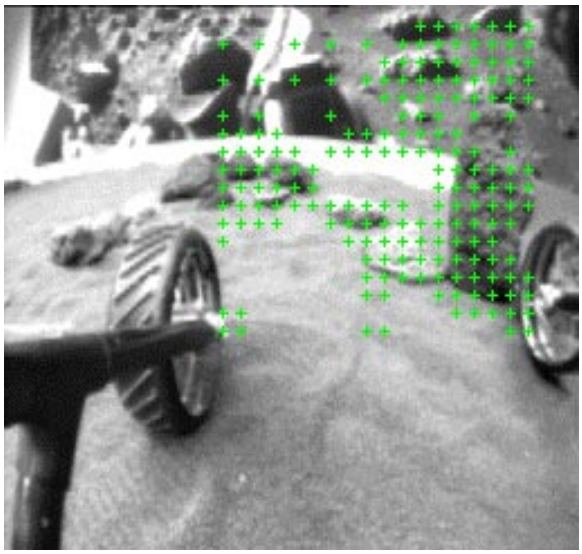

Fig. 5. Points of interest, Level 8.

Figures 7 and 8 offer some comparison between the density of the final range maps generated, between Level 4 and Level 1.

2.1.5. Adaptive Range Map Generation

Summary We can thus summarize the procedure of adaptively generating the range map based on texture and "danger" distance as follows:

1. Obtain stereo image pair (256 x 243, 8 bit grayscale)
2. Compute wavelet transform (2D Harr basis) using the procedure given by [24].
3. Set initial stepsize ($Stepsize_i = 16$) for iteration $i = 0$ and set convergence variable MIN_I
4. Search for points of interest using equation (1) between some max and min row and column locations known to probably contain points of interest.
5. Correlate points using equation (1) and the fundamental matrix.
6. Compute subpixel matches (optional, recommended).
7. Compute 3D locations using triangulation.
8. Perform **Adaptive Mesh Refinement** (see below) on "dangerous" points.
9. $i = i/2$
10. If ($i > MIN_I$) goto 5, else end.

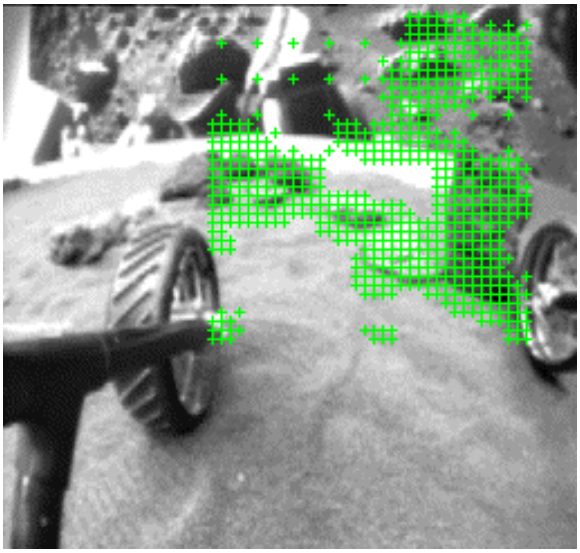

Fig. 6. Points of interest, Level 4.

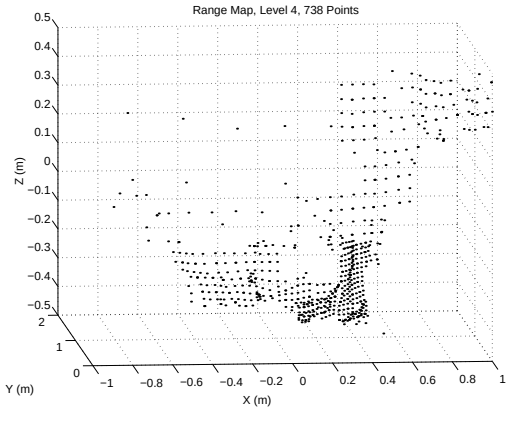

Fig. 7. Generated terrain map (Level 4).

Adaptive Mesh Refinement:

For each point (x,y) preserved as interesting from iteration i:

1. Search for points of interest using equation (1) between $x \pm Stepsize_{i-1}, y \pm Stepsize_{i-1}$ using $(Stepsize_{i-1})/2$.
2. Add any points found to running list of new interesting points.
3. $Stepsize_i = (Stepsize_{i-1})/2$

2.1.6. Timing and Performance

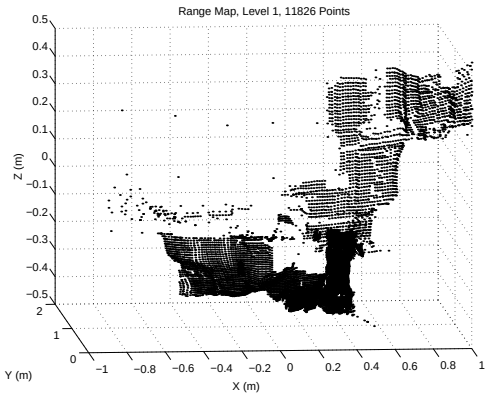

Fig. 8. Generated terrain map (Level 1).

We have time tested the adaptive range map generation algorithm using the 171 image pairs gathered during our first autonomous navigation exercise. At Level 4, the maximum time required to complete range map generation (AND path selection, as described in section 5.1) was 17 seconds (wall clock time) on a Sparc20, running as a user process, with an average time of approximately 9 seconds. At Level 8, the maximum time was approximately 5 seconds (wall clock time) on the same system, with an average time of approximately 2.5 seconds. The same code timed on a Pentium 166 Mhz processor took 7.5 seconds at Level 4, and 1.9 seconds at Level 8, as measured by the profiler under Microsoft Visual C++. Computation time is linear in the number of points computed.

The strength of the algorithm lies in the following features:

- Range maps are computed selectively, only for texturally interesting points, which automatically eliminates those areas where no correlation or an incorrect correlation would be found.
- Range maps are computed adaptively, only for points which are close enough to be accurately located and of possible near-term danger to the rover.
- De-warping and registering the image pairs is not required, as the epipolar geometry is used explicitly to find matching points.
- Easily implementable.
- Low memory requirements.

2.2. Vision-Based State Estimation

At each iteration through our navigation algorithm, we compute a range map, which can be used to identify obstacles in the field of view of the rover. We can also register these successive range maps such that they are all in a single frame of reference, effectively computing a global world map of the environment through which the rover is traversing. The process of registering these maps also has the advantage of providing an estimate of the rover's current state (position and orientation relative to its initial frame of reference).

2.2.1. Extraction of Inter-Frame Rotation and Translation

The algorithm we have implemented for the registration of two range maps is described in [27] and [29]. Here, a brief overview of the algorithm is presented. The interested reader is referred to [27] and [29] for additional details.

This iterative registration process minimizes a performance criteria, $\mathcal{F}$, given by

$$\mathcal{F}(\mathbf{R},\mathbf{T}) = \frac{1}{\sum_{i=1}^{m} p_i} \sum_{i=1}^{m} p_i d^2(\mathbf{R}\mathbf{x_i} + \mathbf{T}, \mathbf{S}') \quad (6)$$

which is related to the difference in distance between the two clouds of 3D range points. In equation (6), $\mathbf{R}$ represents the rotation between the two range maps and $\mathbf{T}$ is the translation between the two range maps. Also in equation (6), the variable p_i is 1 if the point $\mathbf{x_i}$ is matched to a point on the surface $\mathbf{S}'$ defined by the second set of 3D range points, and 0 otherwise, relating to points which are visible only from one frame or the other. The variable m refers to the number of points in the first terrain map, and the function $d(\mathbf{x}, \mathbf{S}')$ is defined by

$$d(\mathbf{x}, \mathbf{S}') = \min_{j \in \{1,\ldots,n\}} d(\mathbf{x}, \mathbf{x}'_\mathbf{j}) \quad (7)$$

where $d(\mathbf{x}, \mathbf{x}')$ is the Euclidean distance between two points.

We are then able to find a set of closest point matches between the two frames where the distances between the two points in a matched pair is less than an adaptively computed maximum distance tolerance D^I_{max}. The search for closest points is accomplished through the use of a *k-dimensional binary search tree* [27] for speed.

In each iteration we adapt the maximum distance tolerance based on the mean μ and standard deviation σ of the distance between the points in each set of matched pairs which satisfied the maximum distance tolerance D^{I-1}_{max} from the previous iteration.

We then adaptively specify a new D^I_{max} based on the following criterion:

if $\mu < \mathcal{D}$
$D^I_{max} = \mu + 3\sigma$
else if $\mu < 3\mathcal{D}$
$D^I_{max} = \mu + 2\sigma$
else if $\mu < 6\mathcal{D}$
$D^I_{max} = \mu + \sigma$
else
$D^I_{max} = \xi$

We reject any points which do not satisfy the newly computed D^I_{max} distance tolerance, and use the remaining matched pairs to compute the motion between the two frames. To compute the motion between the two frames, a dual quaternion methodology [25] [27] is used to solve optimally for the rotation and translation between the two frames simultaneously.

Once the motion between the two frames has been determined, the computed motion is applied to the first frame. We then iterate, finding the next set of closest point matches, updating the point matches using the statistically computed distance tolerance D^I_{max}, computing the motion between the two frames, and applying that motion to the first set of points, until the change in the computed motion reaches a lower threshold, or the algorithm reaches the maximum allowed number of iterations.

One variable referenced above is $\mathcal{D}$, which is computed to be the average distance between points and their closest neighbors in a terrain map, and is effectively related to the resolution of the terrain map data. The second variable, ξ, is computed from the point match set distance histogram. The reader is referred to [27] for a more thorough explanation of $\mathcal{D}$ and ξ, and their implications to the convergence of the algorithm.

2.2.2. Experimental Validation

We now present experimental validation of the algorithm described above, using terrain maps generated by the adaptive range map generation algorithm presented in Sections 2.1.1 - 2.1.6. Figure 9 shows a terrain map generated in one position by the rover. Figure 10 shows a terrain map generated by the rover after a nominal 10° counterclockwise rotation. Applying the range map registration algorithm to the above two range maps, with the following initial rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$:

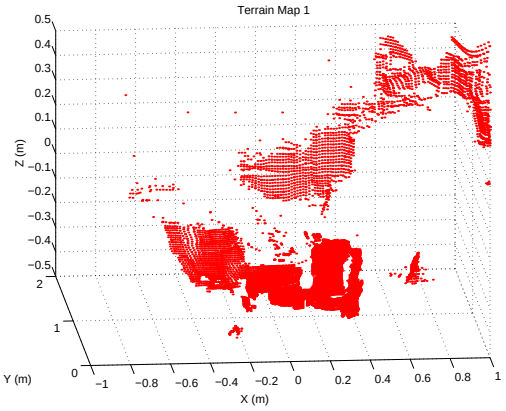

Fig. 9. Range map in position 1; 12,634 points.

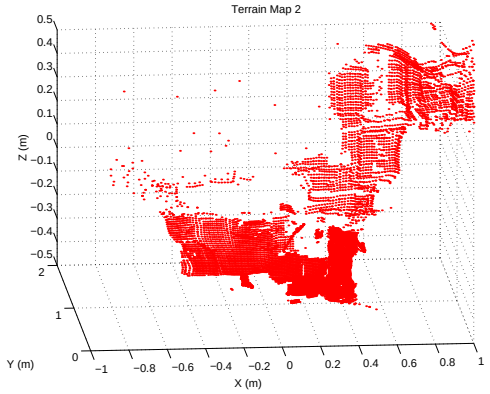

Fig. 10. Range map in position 2; 12,464 points.

$$\mathbf{R_{initial}} = \begin{bmatrix} 0.984807 & -0.173648 & 0.000000 \\ 0.173648 & 0.984807 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

$$\mathbf{T_{initial}} = \begin{bmatrix} 0.000000 & 0.000000 & 0.000000 \end{bmatrix}$$

The final computed rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$ is computed to be:

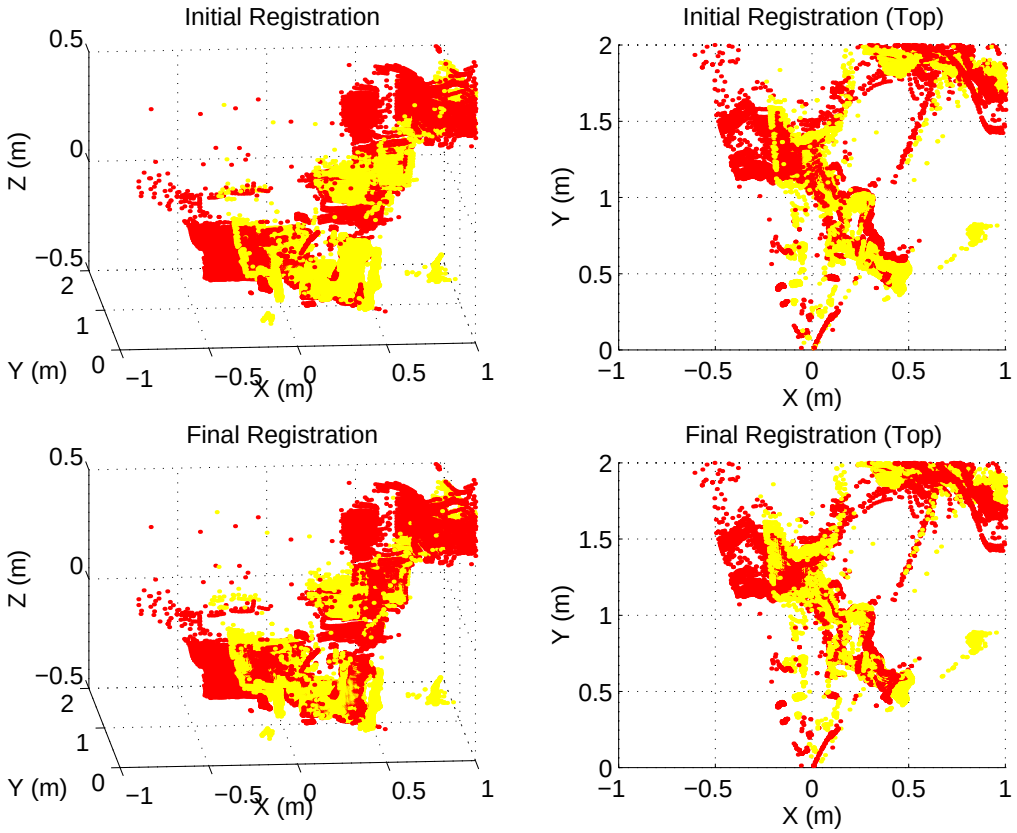

Fig. 11. Initial Registration vs. Final Registration; Light Points: Terrain Map 1, Dark Points: Terrain Map 2; Computed Rotation: 8.26 degrees counterclockwise

$$\mathbf{R_{final}} = \begin{bmatrix} 0.989621 & -0.135732 & -0.047190 \\ 0.136498 & 0.990550 & 0.013388 \\ 0.044927 & -0.019690 & 0.998796 \end{bmatrix}$$

$$\mathbf{T_{final}} = \begin{bmatrix} -0.044082 & -0.026828 & 0.013626 \end{bmatrix}$$

The vehicle rotation computed from $\mathbf{R_{final}}$ results in an 8.26° counterclockwise rotation which corresponds well with the observed motion of the rover, and with the commanded rotation of 10.0° counterclockwise. Figure 11 shows the initial registration between the two terrain maps, after the application of $\mathbf{R_{initial}}$ and $\mathbf{T_{initial}}$ to terrain map 1, and the final registration between the two terrain maps, after the application of $\mathbf{R_{final}}$ and $\mathbf{T_{final}}$ to terrain map 1.

Recent experimental trials indicate that the initial rotation matrix and translation vector can be in error by as much as 50% before an incorrect minimum is reached. Also, the accuracy of the map registration technique is heavily dependent on the quality of the range map which in turn is a function of the stereo camera calibration. For the field of view of the stereo camera pair, range maps are accurate to within 1-2 centimeters. A formal error analysis will be conducted to determine how this range uncertainty propagates into the final rotation matrix and translation vector.

3. State Estimation

Due to various factors, the motion estimate that is produced by the map registration technique presented in the previous section may be in error. These factors will include errors in the camera calibration from which the range maps are generated and the finite precision for which the performance criteria in equation (6) can be computed. Therefore, the vision estimate provides one means for establishing the position and orientation of the rover throughout its traverse. Likewise, the nominal wheel odometry can be utilized to estimate the rover's location. However, like the vision-based state estimates, the dead-reckoned rover location estimates will be in error due to numerous factors including wheel slip in soft terrain, incorrectly-modeled rover kinematics, and inaccuracies due to traverse over obstacles by the rover's rocker-bogie suspension. This section describes the implementation of an extended Kalman filter methodology [2] to the problem of fusing together the rover state estimates as produced by the vision system and dead-reckoning.

3.1. Dead-Reckoned Estimates

The nominal estimates of the rover motion due to wheel odometry alone (e.g. dead-reckoning) can be described by the following state equations:

$$\begin{aligned}\frac{d\mathbf{x}(\alpha)}{d\alpha} &= \begin{bmatrix} \frac{dX(\alpha)}{d\alpha} \\ \frac{dY(\alpha)}{d\alpha} \\ \frac{d\phi(\alpha)}{d\alpha} \end{bmatrix} = \begin{bmatrix} R\cos\phi(\alpha) \\ R\sin\phi(\alpha) \\ \frac{R}{B}u \end{bmatrix} + \mathbf{w}(\alpha) \\ &= \mathbf{f}(\mathbf{x}(\alpha), u) + \mathbf{w}(\alpha) \end{aligned} \tag{8}$$

where $\mathbf{x} = [X\ Y\ \phi]^{\mathrm{T}}$ represents the in-plane position and orientation of the rover relative to some global reference frame, R is the nominal wheel radius, and B is half the distance between the wheel base. The independent variable, α, is defined as

$$\alpha = \frac{\theta_r + \theta_l}{2} \tag{9}$$

where θ_r and θ_l are the absolute wheel rotations of the right and left wheels, respectively. The control variable, u is defined as

$$u = \frac{d\theta_r - d\theta_l}{d\theta_r + d\theta_l} \tag{10}$$

where $d\theta_r$ and $d\theta_l$ are the differential wheel rotations of the right and left wheels, respectively. Finally, $\mathbf{w}(\alpha)$ represents a Gaussian-distibuted, white noise process known as the process noise associated with the state equations. The process noise accounts for random errors such as wheel-slippage.

For a skid-steered rover, the state equations given in equation (7) are valid for all maneuvers except turn-in-place for which the independent variable, α, goes to zero and the state equations become singular. To handle this special case, α is re-defined as

$$\alpha = \frac{\theta_r - \theta_l}{2} \tag{11}$$

which results in the following state-equations for a turn-in-place maneuver

$$\begin{aligned}\frac{d\mathbf{x}(\alpha)}{d\alpha} &= \begin{bmatrix} \frac{dX(\alpha)}{d\alpha} \\ \frac{dY(\alpha)}{d\alpha} \\ \frac{d\phi(\alpha)}{d\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{R}{B} \end{bmatrix} + \mathbf{w}(\alpha) \\ &= \mathbf{f}(\mathbf{x}(\alpha)) + \mathbf{w}(\alpha) \end{aligned} \tag{12}$$

In either case, with the real-time sensing of the wheel rotations via rotary encoders, the above state equations can be numerically integrated (e.g. propagated) in real time thereby producing the dead-reckoned estimates of the rover motion.

Likewise, the estimation error covariance matrix, $\mathbf{P}(\alpha)$, associated with the Kalman filter is propagated in real-time by the following

$$\frac{d\mathbf{P}(\alpha)}{d\alpha} = \mathbf{F}(\alpha)\mathbf{P}(\alpha) + \mathbf{P}(\alpha)\mathbf{F}(\alpha)^{\mathrm{T}} + \mathbf{Q} \tag{13}$$

where $\mathbf{F}(\alpha)$ is the Jacobian of the state equations and $\mathbf{Q}$ is the covariance matrix associated with the process noise. The process noise covariance matrix is assumed to be a diagonal matrix, i. e. $\mathbf{Q} = \mathrm{diag}[Q_{XX}, Q_{YY}, Q_{ZZ}]$, and, qualitatively, represents the confidence placed in the dead-reckoned estimates of the state.

3.2. Kalman Filter Update via Vision

As presented in section 2, the inter-frame rotation and translation of the rover can be determined (to within some level of accuracy) by resolving two successive vision-based range maps. As an output from this map registration technique, we will consider the in-plane rotation and translation

of the rover as a measurement of the rover state that will be fused together with the dead-reckoned state estimates via the Kalman filter approach. Therefore, the measurement of the rover state is

$$\begin{aligned}\mathbf{z}(\alpha_i) &= \begin{bmatrix} X(\alpha_{i-1}) + \Delta X(\alpha_i) \\ Y(\alpha_{i-1}) + \Delta Y(\alpha_i) \\ \phi(\alpha_{i-1}) + \Delta\phi(\alpha_i) \end{bmatrix} + \mathbf{v}(\alpha_i) \\ &= \mathbf{h}(\mathbf{x}(\alpha_i)) + \mathbf{v}(\alpha_i) \end{aligned} \quad (14)$$

where ΔX, ΔY, and $\Delta\phi$ are computed directly from $\mathbf{R}$ and $\mathbf{T}$ as detailed in section 2.2. Also, in equation (13), α_i represents the value of α at which time a stereo pair is acquired and the map registration algorithm is completed and α_{i-1} is the value of α at which the previous vision-based update to the Kalman filter was computed.

Therefore, the update to the rover state is given by

$$\begin{aligned}\hat{\mathbf{x}}(\alpha_i) &= \hat{\mathbf{x}}(\alpha_i|\alpha_{i-1}) + \\ &\quad \mathbf{K}(\alpha_i)[\mathbf{z}(\alpha_i) - h(\hat{\mathbf{x}}(\alpha_i|\alpha_{i-1})] \end{aligned} \quad (15)$$

where the Kalman gain, $\mathbf{K}(\alpha_i)$, is

$$\begin{aligned}\mathbf{K}(\alpha_i) &= \mathbf{P}(\alpha_i|\alpha_i - 1)\mathbf{H}^{\mathrm{T}} * \\ &\quad [\mathbf{H}\mathbf{P}(\alpha_i|\alpha_i - 1)\mathbf{H}^{\mathrm{T}} + \mathbf{R}]^{-1} \end{aligned} \quad (16)$$

and where $\hat{\mathbf{x}}$ refers to the Kalman filtered estimate of the rover state. In equation (15), $\mathbf{H}$ is the Jacobian of the measurement equation given by (13) which for this case is the 3 x 3 identity matrix, $\mathbf{I}$. Also, the measurement noise covariance matrix is denoted by $\mathbf{R}$ and is assumed to be a diagonal matrix, i. e. $\mathbf{R} = \mathrm{diag}[R_{XX}, R_{YY}, R_{ZZ}]$, and, qualitatively, represents the confidence placed in the vision-based estimates of the state.

The estimation error covariance matrix is updated as follows

$$\mathbf{P}(\alpha_i) = (\mathbf{I} - \mathbf{K}(\alpha_i)\mathbf{H})\mathbf{P}(\alpha_i|\alpha_{i-1}) \quad (17)$$

where, in the above equation and in equation (15), $\mathbf{P}(\alpha_i|\alpha_{i-1})$ represents the propagated estimation error covariance matrix produced by the integration of equation (13). Likewise, in equation (14), $\hat{\mathbf{x}}(\alpha_i|\alpha_{i-1})$ represents the propagated state estimates produced by integrating forward the state equations given either by equation (7) or by equation (11).

The performance of this state estimator is presented in section 5. As with any Kalman filter application, the performance of the filter is greatly affected by the assumed process noise covariance matrix, $\mathbf{Q}$, and the measurement noice covariance matrix, $\mathbf{R}$. This is especially true for the rover case where, for example, the type of terrain (e.g. soft soil, hard-pack ground, etc.) can change the quality of the state estimates produced by dead-reckoning. Therefore, in the results presented in this paper, the state-estimate gains are defined as

$$G_X = \sqrt{\frac{R_{XX}}{Q_{XX}}},\ G_Y = \sqrt{\frac{R_{YY}}{Q_{YY}}},\ G_\phi = \sqrt{\frac{R_{\phi\phi}}{Q_{\phi\phi}}}$$

so that the relative weight given to the dead-reckoned estimates with respect to the vision-based estimate can be varied.

4. Experimental Platform

4.1. Hardware Description

The algorithms described in this paper were experimentally verified using NASA/JPL's Lightweight Survivable Rover (LSR-1) platform which utilizes a six wheeled, skid steered, rocker-bogie rover design. MicroArm-1, mounted to LSR-1, is a 4 degree of freedom robotic manipulator arm, 0.7 m at full extent, driven by 1-inch-pound torque capability piezoelectric ultrasonic motor actuators through 200:1 harmonic gear reductions. The powered multifunction end effector has available a close distance color imaging camera, a rotary abrading tool, and a clam-shell scoop/gripper. LSR-1 is pictured in figure 12 and MicroArm-1 is pictured in figure 13 [23].

4.2. Sensor Suite

LSR-1 is outfitted with a black and white stereo CCD pair with a 10 cm baseline, mounted directly beneath MicroArm-1. Each camera is equipped with a 120° field-of-view lens. The stereo pair as mounted on LSR-1 may be seen in figure 14.

This camera pair was shown several black calibration fixtures having white dots of varying diameters in 34 different positions to provide input data for a least-squares calibration technique which attempts to generate a camera model com-

pensating for radial lens distortion [10]. Given the large field-of-view of our cameras, radial lens distortion was quite significant. An example pair of calibration images may be seen in figure 15.

Other than the stereo camera pair (or dead reckoning), LSR-1 has no method by which it may attempt to determine it's change in rotation or translation, as it is not current equipped with wheel encoders or gyro/accelerometers. However, nominal rover translation and rotation of LSR-1 is known via the bang-bang control of the rover wheels under a no-wheel-slip assumption.

4.3. Computing Environment

The computing environment for the LSR-1 platform is currently a hybrid SUN Sparc-20/Silicon Graphics Indigo-2/VMEbus architecture. Computation is performed on the Sparc-20 using C and Java, with low level commands being sent to the LSR-1 rover via the VMEbus using socket connections. Frame grabbing is accomplished using the Indigo-2 video hardware. A Java-based graphical user interface is provided to remotely initiate autonomous routines, as well as display images and data as collected by the rover to the remote scientist/operator. A block diagram of the current

Fig. 12. Lightweight Survivable Rover (LSR-1).

Fig. 13. MicroArm-I.

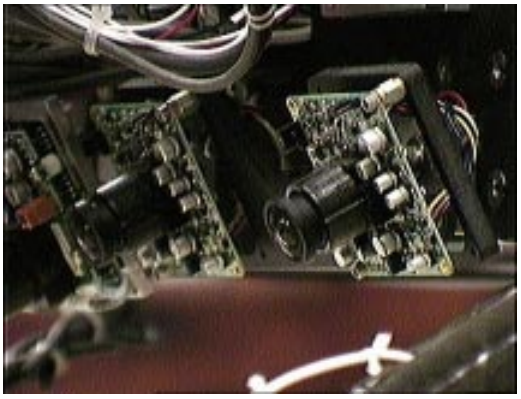

Fig. 14. Stereo CCD cameras.

Fig. 15. Sample stereo calibration pair.

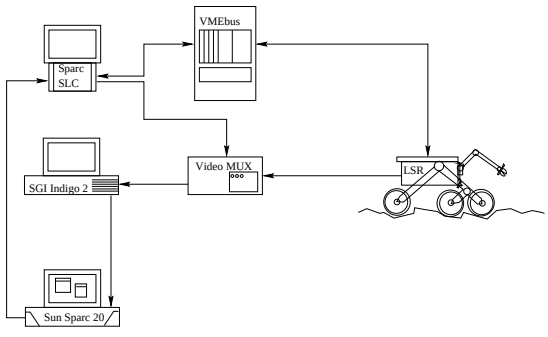

Fig. 16. System setup.

computing environment for the LSR-1 rover is pictured in figure 16.

5. Results

We now present the results for the two navigation modes in which our rover currently may operate: reactive obstacle avoidance and potential flow based path planning for navigation to a user designated goal. All of the results presented in this section were carried out within the rover pit in JPL's Planetary Robotics Laboratory. The 9.1 meter x 4.5 meter rover pit is filled with a soft sand material and is populated with a variety of rock type and sizes. Throughout each run, the rover experiences significant wheel slip, including the occasional wheel hang-up on an obstacle. The rover also traverses over small rocks whose height is determined to be less than one rover wheel diameter. This environment is very indicative of the type of terrain that would be encountered by a rover during Martian surface operations.

5.1. Reactive Obstacle Avoidance

5.1.1. Path Selection Methodology

Once we have adaptively computed the range map, we make use of the terrain information to perform obstacle avoidance. To do so, we sum the number of terrain points encountered when sweeping a box having an outline of the frontal area of the rover through 1.5 meters along the Y axis (directly ahead of the rover). We choose 1.5 meters as our lookahead distance because of the accuracy limitations imposed by low resolution images taken through 120° field of view lenses.

The path having both the least number of encountered points and also the least amount of required rotation is chosen. If the absolute value of the rotation requested is less than 5 degrees, then the rover proceeds forward for 5 seconds (approximately 20 cm). If the rotation is greater than +5 degrees, the rover rotates clockwise for 5 seconds, otherwise the rover rotates counterclockwise for the same period of time, which corresponds to a rotation of approximately 10 degrees. We find that this technique gives us adequate obstacle avoidance performance.

Using only current information and the above strategy may lead to oscillatory behavior during obstacle avoidance behaviors. The rover thus maintains a history of its past executed motions and initiates an oscillation-breaking maneuver (20 degree rotation) if it detects oscillatory behavior.

5.1.2. State Estimate During Reactive Obstacle Avoidance

As can be seen in figure 17, the Kalman filtered vehicle state estimate is quite good after approximately a 6 meter traverse. Also in figure 17, the path labeled "vision" refers to the rover state produced by the vison-based estimate of rover inter-frame rotation and translation alone without knowledge of wheel odometry. The final error between the filtered state and the ground truth is approximately 0.22 meters, with a maximum absolute error over the whole traverse of 0.64 meters, and a mean of 0.38 meters. The vehicle made 79 iterations through the reactive obstacle avoidance algorithm, making 20 right turns (10° nominal), 11 left turns (10° nominal), and 48 forward motions (20 cm nominal). Figure 18 details the position error of the Kalman filtered estimate with respect to the ground truth. The gains used in the Kalman filter were $G_X = 1.9$, $G_Y = 1.9$, $G_\phi = 1.2$.

The ground truth in all trials is computed by selecting the cross shaped target on the top of the rover in the view provided by a 120° field-of-view camera mounted on the ceiling of the rover workspace. This so called "sky camera" has been calibrated by presenting it with the fixture pictured in figure 15 in multiple locations covering the floor of the workspace, and then fitting a third degree polynomial to the calibration points, relating pixel location in the image with ground location. A separate curve was fitted for the inverse relationship. The ground truth is accurate to within approximately 5 cm.

5.2. Potential Flow-Based Path Planning

As an alternative to simply performing reactive obstacle avoidance, we can also navigate to a user designated goal and avoid (either autonomously

located or user designated) obstacles along the way. To do this, we have implemented a real-time path planner based on potential flow methodologies [7]. During each iteration of the navigation algorithm, this path-planner computes the path from the rover's current estimated position to the desired goal. Currently, we limit path planning to 2D and obstacles to be limited to cylinders or ellipses at some specified angle, although arbitrarily shaped objects may be modeled using panel-approximations to the object's boundary.

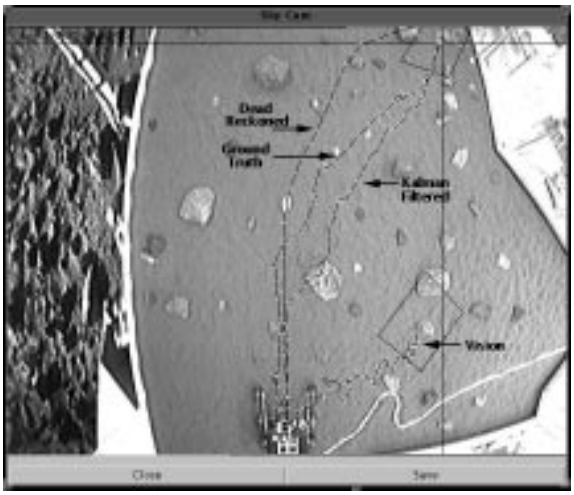

Fig. 17. State estimates during reactive obstacle avoidance run.

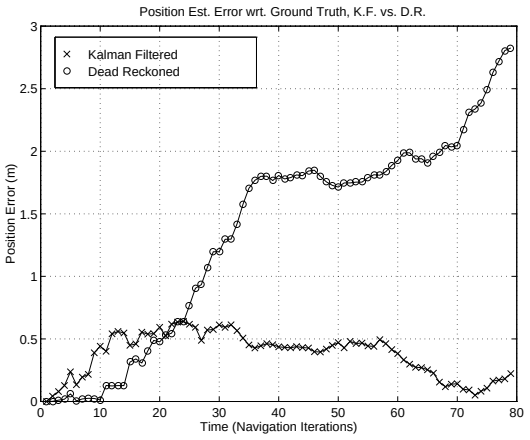

Fig. 18. Position error (m); Kalman filtered vs. dead reckoned; reactive run.

5.2.1. Obstacle Modeling

Here we present a brief treatment of the potential flow based path planning methodology described more fully in [7]. The interested reader is referred there for more details.

If we consider the rover workspace as the complex plane, the potential for a plane flow at an angle α with the real axis is given by

$$w = -Uze^{-i\alpha} \ . \tag{18}$$

The potential for a circular cylinder centered at $z = X_0 + iY_0$ with radius r in a plane flow with velocity U at an angle α with the real axis is given by

$$w = -U(ze^{i\alpha} + \frac{r^2 e^{i\alpha}}{z - z_0}) \tag{19}$$

Finally, the flow around an ellipse with major axis a and minor axis b may be obtained using a conformal mapping as given by

$$z = \xi + \frac{r^2}{4\xi} \tag{20}$$

Applying the conformal mapping given by equation (20) to equation (19), we obtain the equation for a cylinder in the ξ plane, as may be seen in equation (21).

$$w = -U(\xi e^{-i\alpha} + \frac{(a+b)^2 e^{i\alpha}}{4\xi}) \ . \tag{21}$$

If we then solve for ξ, we obtain the following

$$\xi = \frac{1}{2}(z \pm \sqrt{z^2 - r^2}), r^2 = a^2 - b^2 \tag{22}$$

whose solution describes the velocity field around the ellipse.

5.2.2. Potential Construction

We define the external plane flow using X_f and Y_f as the (X, Y) goal position, and X and Y are the rover's current position.

$$w = -Uze^{-i\alpha}, \alpha = \tan^{-1}(\frac{Y_f - Y}{X_f - X}) \tag{23}$$

An approximate harmonic potential solution to the continuous potential flow is achieved by the superposition of the closed form potentials for the plane flow and all of the individual obstacles added to the flow. More details on the construction of this harmonic potential are provided in [7].

5.2.3. Rover Motion

Once the harmonic potential is constructed, the u and v components of the flow can be found using equation (24).

$$\begin{aligned} u = \frac{\partial \phi}{\partial X} = \frac{\partial \psi}{\partial Y} = -\Re(\frac{dw}{dz}), \\ v = \frac{\partial \phi}{\partial Y} = -\frac{\partial \psi}{\partial X} = \Im(\frac{dw}{dz}). \end{aligned} \tag{24}$$

The u and v components of the flow define a vector in the rover workspace. The potential flow based path planning methodology was originally developed for an omni-rover. However, our rover may only move in the direction which it is facing. Thus, we must align the pointing angle of the rover with the current streamline vector. Once the rover has rotated to be nominally aligned with the streamline by making 10° nominal rotations, (in our implementation, the pointing error must be less than 5°), it may proceed forward, in steps of 20 cm nominal.

Currently obstacle locations are defined *a priori* by the user in the global reference frame of the sky camera. However, given a manipulator-mounted or mast-mounted stereo pair on a rover platform, all obstacles could be located either by the user, or located automatically by processing the terrain map generated by this stereo pair, all in the reference frame of the rover. We plan to pursue this avenue in the future.

As a point of note, it is explicitly the ability of the real-time path planner to deal with dynamic environments which allows its application to the context of rover navigation. If all obstacle locations are not known *a priori*, then incremental discovery of obstacles and their addition to the world potential flow requires a path planner capable of dealing with dynamic environments.

Similarly, as the commanded motion from the path planner may not actually be the exact motion achieved, due to wheel slip and numerous other factors, we require a path planner that is capable of dealing with a dynamic environment. This can be seen by assuming that if the rover actually *did* achieve the commanded position exactly, then the obstacles must have moved so as to be in the same relative position with respect to the rover. Thus, the rover must be capable of replanning its path to the goal in real time in a dynamic environment, or in response to its current best position and orientation estimate.

5.2.4. State Estimates During Potential Flow-Based Path Planning

Figure 19 details the traverse of the LSR vehicle to a nearly directly accessable goal, through soft soil-simulant and over medium sized rocks. The final error between the filtered state and the ground truth was 0.63 meters, with a maximum absolute error over the whole traverse of 0.76 meters, and a mean of 0.40 meters. The vehicle made 125 iterations through the potential flow based navigation algorithm, making 41 right turns, 44 left turns, and 40 forward motions. The gains used in the Kalman filter for this first run were $G_X = 1.9$, $G_Y = 1.9$, $G_\phi = 1.2$.

Figure 21 details the traverse of the LSR rover around an obstruction to a goal. The final error between the filtered state and the ground truth was 0.67 meters, with a maximum absolute error over the whole run of 1.09 meters and a mean of 0.59 meters. The vehicle made 69 motions, making 8 right turns, 22 left turns, and 39 forward motions. The gains used in the Kalman filter for the second run were $G_X = 5.0$, $G_Y = 5.0$, $G_\phi = 2.5$.

Table 1 summarizes our results for all of the experimental trials presented in this section.

6. Conclusions

In summary, this paper has shown that the fusing of multiple estimates of a rover's state via an extended Kalman filter framework change can have a profound impact on the quality of the rover's state estimate as a whole, in some cases more than doubling the accuracy of the state estimate with respect to dead-reckoned estimates. Additionally, the paper has presented a viable methodology for the texture-based constructing semi-sparse range maps given computational resource restrictions, especially if it is made adaptive to current ground texture and distances to possible obstacles. Fi-

nally, this paper successfully demonstrated the use of real-time, potential flow based path planning methodologies as applied to complex, dynamic environments with very positive results.

The described algorithms are currently being implemented on NASA's Sample Retrieval Rover (SRR) platform (shown in figure 20) which is a 4-wheeled, skid-steered, split-differential rover design capable of a sustained top ground speed of approximately 35 cm/sec. This rover is dedicated to the development of advance planetary rover technologies. SRR is equipped with wheel encoders, an INS system, and a full vision system. This rover will serve as the platform on which future investigations regarding the development of state estimators that will fuse INS measurements, vision measurements, and wheel odometry using the extended Kalman filter framework described in this paper.

Table 1. State estimation results for all experimental trials.

Trial	Kalman Filtered Max/Mean	Dead Reckoned Max/Mean
Reactive	0.68/0.38 m	2.83/1.34 m
Unobstructed	0.76/0.40 m	2.99/1.38 m
Obstructed	1.09/0.59 m	1.34/0.70 m

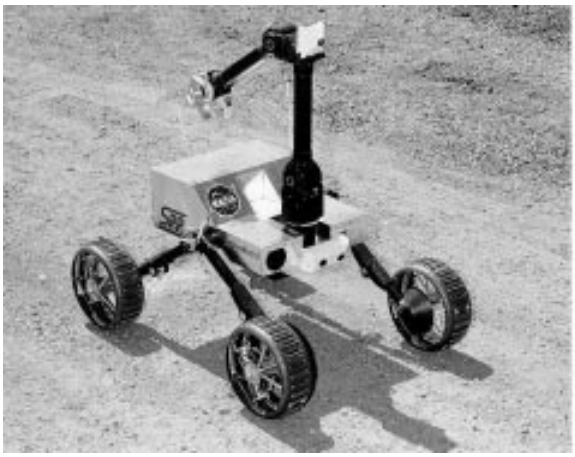

Fig. 20. Sample Return Rover (SRR).

Acknowledgements

The authors would like to acknowledge Hrand Aghazarian, JPL, for many insightful discussions and for his assistance in implementing the described algorithms on the SRR platform. Additionally, we would like to acknowledge Hans Jacob Feder, MIT, for the original version of the potential flow code, and Todd Litwin, JPL, for the original version of the stereo triangulation and camera calibration code. This work has been supported by

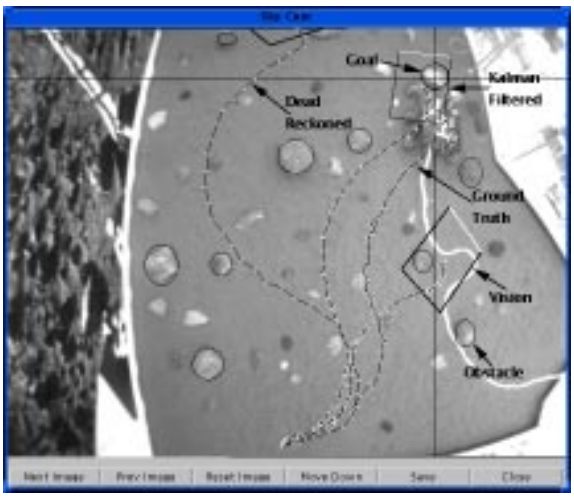

Fig. 19. State estimates during potential flow-based path planning run: unobstructed goal.

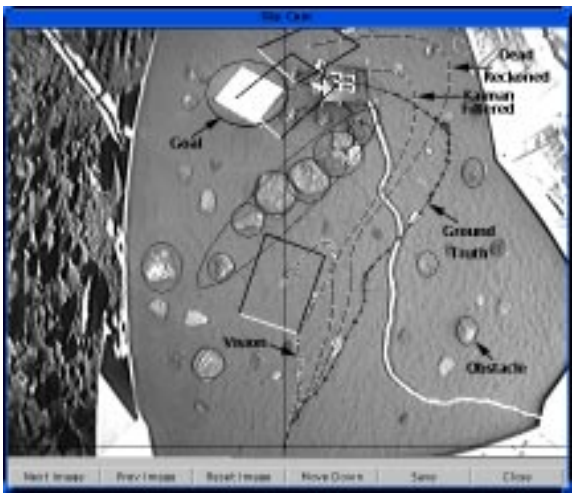

Fig. 21. State estimates during potential flow-based path planning: obstructed goal.

the NASA Telerobotics Technology Program directed by Dave Lavery, NASA Headquarters, and Charles Weisbin, JPL Technology and Applications Program (TAP) Directorate.

References

1. Barshan, B. and Durrant-Whyte, H. F., "Inertial Navigation Systems for Mobile Robots," *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 3, pp. 328-342, June 1995.
2. Baumgartner, E. T., and Skarr, S. B., "An Autonomous Vision-Based Mobile Robot," *IEEE Transactions on Automatic Control*, Vol. 39, No. 3, 493-502, March 1994.
3. Bernard, S. T., Fischler, M. A., "Computational Stereo," Comput. Surv. 14(4):553-572, 1982.
4. Betge'-Brezetz, S., Chatila, R., Devy, M., "Object-based Modelling and Localization in Natural Environments," *IEEE International Conference on Robotics and Automation*, pp. 2920-2927, 1995.
5. Connolly, C. I., Burns, J. B., Weiss, R. "Path Planning Using Laplace's Equation," *Proc. IEEE International Conference on Robotics and Automation*, pp. 2101-2106, 1990.
6. Espinal, F., Huntsberger, T., Jawerth, B., Kubota, T., "Wavelet-Based Fractal Signature Analysis for Automatic Target Recognition," *Optical Engineering*, Vol. 37, No. 1, pp. 166-174, January 1998.
7. Feder, H. J. S., and Slotine, J-J. E., "Real-Time Path Planning Using Harmonic Potentials in Dynamic Environments," *IEEE International Conference on Robotics and Automation (ICRA)*, April 1997.
8. Fua, P., "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, 6(1), Winter 1993.
9. Gennery, D. B., "Visual terrain matching for a Mars rover.", *Proc. International Conference on Computer Vision and Pattern Recognition*, pp. 483-491, San Diego, CA, June 1989.
10. Gennery, D. B., "Least-Squares Camera Calibration Including Lens Distortion and Automatic Editing of Calibration Points," *Calibration and Orientation of Cameras in Computer Vision*, A. Grün and T. Huang, editors, Springer-Verlag, 1993.
11. Hebert, M., Caillas, C., Krotkov, E., Kweon, I.S., Kanade, T. "Terrain Mapping for a Roving Planetary Explorer", *Proc. IEEE International Conference on Robotics and Automation*, pp. 997-1002, 1989.
12. Kim, J-O., Khosla, P., "Real-Time Obstacle Avoidance Using Harmonic Potential Functions," *Proc. IEEE International Conference on Robotics and Automation*, 1991.
13. Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robotics," *The International Journal of Robotics Research*, **5**, No. 1, pp. 90-98, 1986.
14. Lacroix, S., Fillatreau, P., Nashashibi, F., "Perception for Autonomous Navigation in a Natural Environment," *Workshop on Computer Vision for Space Applications*, Antibes, France, Sept. 22-24, 1993.
15. Lain, A., and Fan, J., "Texture Classification by Wavelet Packing Signatures," *IEEE Trans. PAMI*, Vol. 15, No. 11, pp. 1186-1191, November, 1993.
16. Matthies, L., Gat, E., Harrison, R., Wilcox, B., Volpe, R., Litwin, T., "Mars Microrover Navigation: Performance Evaluation and Enhancement," *Autonomous Robots Journal*, special issue on Autonomous Vehicles for Planetary Exploration, Vol. 2(4), pp. 291-312, 1995.
17. Matthies, L., "Dynamic Stereo Vision," Ph.D. Thesis, Computer Science Department, Carnegie Mellon, 1987.
18. Matthies, L., Olson, C., Tharp, G., Laubach, S., "Visual Localization Methods for Mars Rovers Using Lander, Rover, and Descent Imageery," *International Symposium on Artificial Intelligence, Robotics, and Automation in Space* (i-SAIRAS), Tokyo, Japan, July 1997.
19. Olson, C., "Mobile Robot Self-Localization by Iconic Matching of Range Maps," *Proc. of the 8th International Conference on Advanced Robotics*, pp. 447-452, 1997.
20. Quinlan, S., Khatib, O., "Elastic Bands: Connecting Path Planning and Control," *Proc. IEEE International Conference on Robotics and Automation*, **2**, pp. 802-808, 1993.
21. Roberts, B. and Bhanu, B., "Inertial Navigation Sensor Integrated Motion Analysis for Autonomous Vehicle Navigation," *Journal of Robotic Systems*, Special Issue on Passive Ranging For Robotic Systems, Vol. 9, No. 6, pp. 817-842, September 1992.
22. Roumeliotis, S., Bekey, G., "An Extended Kalman Filter for frequent local and infrequent global sensor data fusion," *Proc. SPIE Vol. 3209*, 1997.
23. Schenker, P.S., Baumgartner, E.T., Lee, S., Aghazarian, H., Garrett, M.S., Lindemann, R.A., Brown, D.K., Bar-Cohen, Y., Lih, S., Joffe, B., Kim, S.S., Hoffman, B.D., Huntsberger, T., "Dexterous robotic sampling for Mars *in-situ* science", *Intelligent Robotics and Computer Vision XVI*, Proc. SPIE 3208, Pittsburgh, PA, Oct 14-17, 1997.
24. Stollnitz, E. J., DeRose, T. D., Salesin, D. H., 'Wavelets for Computer Graphics, Theory and Applications," Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.
25. Walker, M. W., Shao, L. and Volz, R. A., "Estimating 3-D location parameters using dual number quaternions," CVGIP: *Image Understanding* 54(3), 358-367.
26. Xiong, Y. and Matthies, L., "Error Analysis of a Real-Time Stereo System," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1087-1093, 1997.
27. Zhang, Z., "Iterative Point Matching for Registration of Free-Form Curves and Surfaces," *International Journal of Computer Vision*, 13:2, 119-152, 1994.
28. Zhang, Z., Deriche, R., Faugeras, O., Luong, Q-T., "A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry," *Institute National De Recherche En Informatique et en Automatique (INRIA)*, Rapport de recherche No. 2273, May 1994.
29. Zhang, Z., "A Stereovision System for a Planetary Rover: Calibration, Correlation, Registration, and Fusion," *Proc. IEEE Workshop on Planetary Rover*

Technology and Systems, Minneapolis, Minnesota, April 23, 1996.

30. Zhang, Z., "On the Epipolar Geometry Between Two Images With Lens Distortion," *Proc. Int'l Conf. Pattern Recognition (ICPR)*, Vol. I, pp. 407-411, Vienna, Aug. 1996.

31. Zhang, Z., Image-Matching software. URL:*http://www.inria.fr/robotvis/personnel/zzhang/softwares.html*

Brian D. Hoffman is a member of engineering staff at Silicon Graphics Inc.

Eric T. Baumgartner is a senior member of engineering staff at NASA's Jet Propulsion Laboratory in Pasadena, CA